Ilidio Mendes Moreira

Base de Dados por Replicação & Sincronização

(Alternativa para instituições com fracos recursos financeiros)

Licenciatura em Matemática - Ramo Informática

Praia, Setembro de 2008

ILIDIO MENDES MOREIRA

BASE DE DADOS POR REPLICAÇÃO & SINCRONIZAÇÃO

(Alternativa para instituições com fracos recursos financeiros)

Licenciatura em Matemática

Ramo Informática

Monografia apresentada para a Obtenção do grau de Licenciatura em Matemática – Ramo Informática pelo Instituto Superior de Educação.

Orientador: *Engenheiro e Mestre Danilo de Sousa Tavares*

ILIDIO MENDES MOREIRA

BASE DE DADOS POR REPLICAÇÃO & SINCRONIZAÇÃO

(Alternativa para instituições com fracos recursos financeiros)

Licenciatura em Matemática

Ramo Informática

Membros de Júri

_____

_____

_____

Praia _____ de _____ de 2008

Agradecimentos

Ao meu orientador que, prontamente, aceitou o desafio apesar de ter uma agenda bastante carregada.

Ao pessoal da Delegacia de Saúde de Santa Catarina e do Hospital Regional Santiago Norte em especial a Dona Leniha e o Sr. Domingos, o Director do Hospital o Sr. Luís Dias, o Enfermeiro Superintendente o Sr. Aniceto Tavares e o então chefe da secção de estatística o Sr. Austolino pelas informações prestadas e pelos documentos facultados.

Um agradecimento especial para a minha esposa, pelos inegáveis apoios que tenho recebido e meus filhos.

Índice

Índices Figuras

Índices de Tabelas

1. Introdução

Hoje, mais de que nunca, se põe a tónica na questão das novas tecnologias sob uma perspectiva de o País poder acompanhar o ritmo do desenvolvimento global de forma sustentável. A reforma da administração pública que o governo iniciou há já alguns anos, requer uma reformatação de todo o processo de gestão das informações justificando alguns marcos já alcançados a saber: a criação do NOSI, a recente assinatura do protocolo com a Microsoft, etc.

Esta dinâmica é almejada para todo o sector de actividade no país. As grandes, médias e pequenas empresas conseguirão engajar-se melhor às exigências de um mercado cada vez mas global, se conseguirem, em tempo útil, decidir sobre politicas mais acertadas com relação aos mercados. Este processo decisório e de resposta às solicitações do mercado, importante para qualquer organização, seja ela pública ou privada, com fins lucrativo ou não, passa, obrigatoriamente, pela capacidade dessas organizações em gerir suas próprias informações e as do mercado. É neste contexto que as bases de dados se revelam de capital importância, constituindo suporte para a automatização dos serviços.

Hoje, no país, já se fala muito em automatização de serviços e em sistemas integrados de gestão, todavia, a sua implementação é muito tímida. Apenas as grandes empresas como as instituições financeiras, as seguradoras, alguns organismos do estado etc. enveredam por esta via pagando, para tal, um elevado custo de comunicação.

Deste modo, que solução para as organizações sem fins lucrativos e as pequenas empresas que, também, querem vencer o desafio da globalização?

Na verdade, muitas organizações temem dar o grande passo, o da modernidade, por desconhecerem alternativas menos caras.

Este trabalho, "Base de Dados por Replicação & Sincronização - Alternativa para instituições com fracos recursos financeiros", tem como um dos propósitos, apresentar a replicação como uma alternativa que, quando adequada à realidade da organização, resulta em incríveis

poupanças de recursos destinados, de outra forma, à comunicação. Desta forma, a replicação prova-se acessível à maioria das organizações.

Dado ao carácter monográfico deste trabalho, passamos a especificar alguns objectivos que se pretende alcançar, quais sejam:

- Compreender bem a teoria de base de dados relacionais;
- Compreender bem o processo de replicação e sincronização das bases de dados – particularizado para o Microsoft Office Access;
- Contrapor as vantagens, desvantagens e situações de opção por uma base de dados centralizada ou por uma distribuída, sincronizada por replicação;
- Adquirir competências a nível de gestão de conflitos de sincronização.

Estes objectivos serão alcançados seguindo a seguinte metodologia:

- Investigação bibliográfica que nos conduzirá a aquisição de uma base teórica razoavelmente sólida, numa fase do trabalho que denominamos "Noções Básicas". Básicas, não só por constituir-se em pré-requisitos, mas também por servir de base para o desenvolvimento da parte prática deste trabalho – Aplicação de Gestão de Informação Estatística da Região Sanitária Santiago Norte (RSSN).

- Entrevistas, numa lógica de compreensão da realidade a modelar, possibilitando a construção do modelo teórico da base de dados a desenvolver na parte prática.

2. Noções básicas

2.1. Bases de dados Relacionais

O modelo de base de dados relacional terá tido seus fundamentos teóricos, primeiramente, em 1969 por Edgar Frank Codd, enquanto investigador da IBM[1]. Este modelo suporta-se em teorias matemáticas, mais especificamente, em elementos da teoria de conjuntos e lógica de predicados. A ideia básica por detrás deste modelo é a de que uma base de dados consiste numa série de tabelas não ordenadas (também designadas relações) que podem ser manipuladas usando operações não procedimentais que retornam tabelas.

Entre a fundamentação teórica, por Codd, e o aparecimento do primeiro sistema baseado no modelo relacional passaram sensivelmente dez anos. Na verdade, só em 1979/80 surge, disponibilizado pela Relational Software Inc. (actualmente designada Oracle Corp.), o primeiro produto com características relacionais - o SGBD Oracle (Sistema de Gestão de Bases de Dados Oracle).

Actualmente existem vários fornecedores de tecnologia relacional no mercado. Entre os SGBDs relacionais mais representativos incluem-se o *CA-OpenIngres* da Computer Associates Inc., o *DB/2* da IBM Corp., o *Informix-OnLine Dynamic Server* da Informix Software Inc., o *Microsoft SQL Server* da Microsoft Corp., o *Oracle Server* da Oracle Corp. e o *Sybase SQL Server* da Sybase Inc.

A estrutura fundamental do modelo relacional é a relação, também designada por tabela e, consequentemente, suas partes constituintes: colunas e linhas. Para os teóricos de base de dados relacionais os termos tabelas, colunas e linhas são substituídos, respectivamente, por relações (que também são chamadas entidades), atributos e registos.

As tabelas, no modelo relacional, representam "coisas" do mundo real. Cada tabela deve representar uma colecção de um mesmo tipo de "coisas". Esta "coisa" ou entidade pode ser

1 - GETZ Ken et all, ACCESS 2000 Developer's Handbook, Vol1, Desktop Edition pag. 94.

um objecto, ou ser um evento do mundo real. A título de exemplo, uma cama, um paciente, e uma consulta são entidades.

O modelo relacional dita que uma linha da tabela – um registo, deve ser única. A duplicação de linhas impossibilita a existência de um endereço único para chegar a um determinado registo. Isso acarreta ambiguidades e muitos problemas para a base de dados.

Para garantir a unicidade dos registos são definidas as *chaves primárias* – uma coluna (*atributo*) ou combinação de colunas que determinam de forma única um registo. Toda a coluna cujo valor é único, é candidata a chave – chaves candidatas, de entre as quais escolhe-se a chave primária. Não existe nenhuma regra absoluta através da qual se declara que uma determinada chave candidata é chave primária. Todavia, Fabian Pascal[2], no seu livro *SQL and Relational Basics*, nota que tal decisão deve ser baseada nos princípios da *minimalidade* (escolher o mínimo de colunas necessária), *estabilidade* (escolher chave que raramente ou nunca se altera) e *simplicidade/familiaridade* (escolher chave que possa ser, ao mesmo tempo simples e familiar para o utilizador).

2.2. Modelo Entidade - Relacionamento

O modelo entidade - relacionamento (MER) é um modelo de dado conceptual de alto nível, cujos conceitos foram projectados para estar o mais próximo possível da visão que o usuário tem dos dados, não se preocupando, contudo, em ilustrar como estes dados estarão armazenados.

O MER pressupõe a adopção dos termos "Entidade" e "Relação" sendo que:

- A "Entidade" corresponde à Tabela e,
- A "Relação" corresponde à ligação entre as Entidades.

Neste modelo, uma Relação, que temos tratado, até agora, como uma Tabela (ou Entidade), passa, por uma questão de conveniência, a ser tratada como a unidade que estabelece a ligação, o relacionamento, entre Entidades.

[2] - GETZ Ken et all, ACCESS 2000 Developer's Handbook, Vol1, Desktop Edition pag. 96.

O MER é utilizado principalmente durante o processo de projecto de banco de dados e concretiza-se numa representação gráfica da realidade em análise.

Os elementos gráficos utilizados neste modelo são:

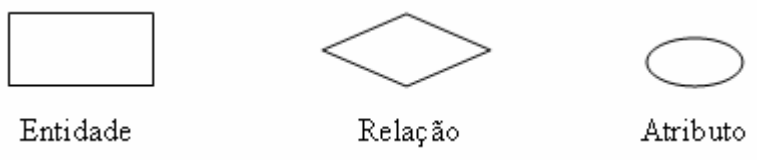


Fig. 2-1 – Elementos gráficos que compõe o MER

Com efeito, se quisermos representar uma relação possível entre médico e paciente, teríamos o seguinte esquema:

Fig. 2-2 – Relação Medico - Paciente

Todavia, este esquema não é, ainda, auto explicativo. É importante mostrar a frequência com que esta relação se dá – *Cardinalidade de Relacionamentos*: relacionamento de cardinalidade 1:1 ou, simplesmente, relacionamento 1:1; relacionamento 1:n ou n:1 e relacionamento n:m. Com efeito, a realidade fica melhor representada, se acrescentada, ao esquema, elementos que respondem questões como: *"um médico consulta quantos pacientes"* ou *"um paciente é assistido por quantos médicos"*. Se quisermos que o esquema anterior reflecte que *um médico consulta vários pacientes*, então a representação seria:

Fig. 2-3 – Relação 1:n

Analogamente, se quiséssemos um esquema que reflectisse a situação em que, um paciente é assistido por vários médicos, teríamos o seguinte:

Fig. 2-4 – Relação n: 1

Quando as situações anteriores têm lugar simultaneamente, está-se perante um relacionamento do tipo n:m, representado graficamente da seguinte forma:

Fig. 2-5 – Relação n:m

A representação gráfica é complementada com a inclusão dos atributos de cada entidade, destacando as chaves primárias (assinaladas com um sublinhado). Fica o modelo, assim, pronto para ser implementado (*fig.2-6*).

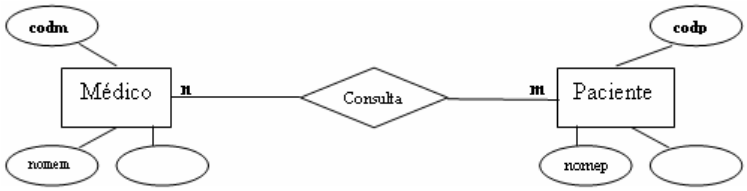
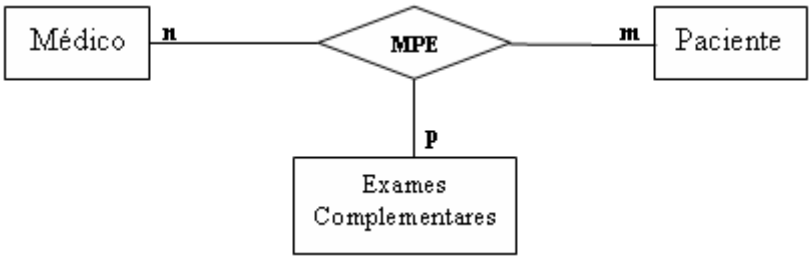


Fig. 2-6 – Modelo ER quase completo

Um outro aspecto que devemos ter em conta, também, é o número de tipo de entidades que participam de um relacionamento – *Grau de Relacionamento*. No exemplo da figura anterior temos um relacionamento binário, pois participam duas entidades, Médico e Paciente. A partir de grau 3 (relacionamento ternário), a compreensão torna-se cada vez mais complexa e a dificuldade de se desenvolver uma relação correctamente torna-se maior. No entanto, isto não nos impede de deixar, aqui, pistas para o tratamento de relacionamentos com grau superior a dois - que passamos a chamar de *Relacionamentos Múltiplos*.

O tratamento das relações múltiplas passa pela sua decomposição em relacionamentos de grau dois. Um relacionamento binário, decorrente, e convenientemente escolhido, é tomado como uma única entidade. Esta, então, relacionando com uma terceira entidade, conforma uma relação ternária. Para elucidar, vejamos o exemplo do relacionamento entre o Médico, o Paciente e os Exames Complementares:



MPE: Relação estabelecida entre Medico, Paciente e Exames Complementares.

Fig. 2-7 - Diagrama de uma relação ternária

Os exames complementares só são feitos ao paciente, a pedido do médico que o terá atendido em consulta. O diagrama corresponde a uma relação ternária cuja implementação requer decomposição que passamos a ilustrar na figura seguinte:

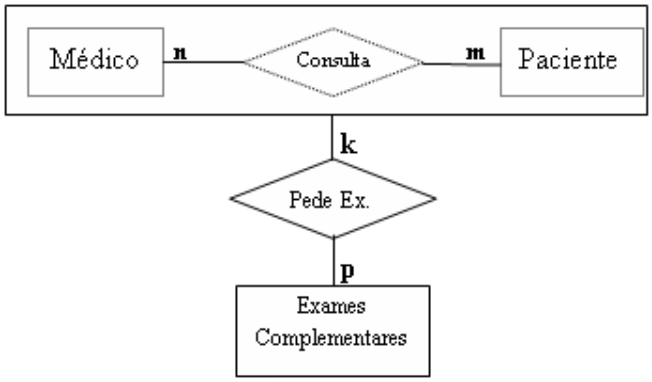


Fig. 2-8 - Diagrama que ilustra um agregado de entidades

O conjunto formado pelas entidades Médico e Paciente é tomado, na sua relação, como uma Entidade (denominado Agregado).

Agregação é pois, uma abstracção por meio da qual relacionamentos são tratados como uma entidade de nível superior.

2.3. Normalização da base de dados

Um dos grandes desafios a vencer aquando da estruturação de uma base de dados é o da minimização da redundância e consequente inconsistência das informações. A superação destes desafios influi directamente na construção de uma estrutura que ocupa menos espaço no disco bem como na maior performance no respeitante ao tempo de processamento e manipulação dos dados. Uma das vias para se conseguir tal resultado é a ***Normalização.***

A Normalizar consiste na decomposição de uma tabela original em "um conjunto de tabelas conceptualmente relacionadas entre si"[3] com objectivo de minimizar redundância da informação. Usualmente, destacam-se três[4] níveis de normalização quais sejam: A 1ª Forma Normal (1FN), a 2ª Forma Normal (2FN) e a 3ª Forma Normal (3FN).

[3] Loureiro Henrique, ACCESS Macros & VBA – Curso Completo, ed. FCA Abril 2006
[4] Actualmente estão fundamentadas cinco formas normais (mais o Boyce/Codd Normal Form) mas, na prática, são mais utilizadas as três primeiras.

Por definição, se uma tabela ou relação está na 1FN, então todos os valores das colunas são designados de valores atómicos, o que significa que não contem repetição. Por exemplo:

Seja A, a estrutura de uma entidade (tabela) constituída pelos seguintes atributos: *cod_consulta, data_consulta, cod_paciente, nome_paciente, residência_paciente, cod_exame, descriacao_exame, (estrutura que passamos a representar por* A(*cod_consulta, data_consulta, cod_paciente, nome_paciente, residência_paciente, cod_exame, descriacao_exame)).* Esta entidade guarda, desnecessariamente, informações repetidas. De facto, numa consulta pode ser feito vários exames e, para cada exame guardada repete-se todas as outras informações do paciente. Para evitar tal repetição desdobremos a tabela A em duas, A' e A'' de seguinte forma:

A' (*cod_consulta, data_consulta, cod_paciente, nome_paciente, residência_paciente) e*

A'' (cod_consulta, cod_exame, descriacao_exame). A tabela A ficou, deste modo, na primeira Forma Normal -1FN.

A passagem da 1FN para 2FN começa com a identificação das chaves (simples e compostas) passando pelo reconhecimento das Dependências Funcionais ou seja, identificação de colunas cujos valores dependem directamente das colunas-chave. A partir de cada dependência funcional cria-se uma nova tabela. Estas novas tabelas apresentam-se na 2FN. Por exemplo, a coluna *descricao_exame* depende directamente do *cod_exame,* dai que a tabela A'' se desdobra em duas outras reduzindo a tabela A à 2FN, com os resultados:

A'(*cod_consulta, data_consulta, cod_paciente, nome_paciente, residência_paciente)*

A_1'' (cod_consulta, *cod_exame)*

A_2'' (cod_exame, *descriacao_exame)*

Para chegar à Terceira Forma Normal (3FN), vamos determinar, em cada tabela na 2FN, quais as colunas cujos valores dependem directamente das colunas que são chaves. Esta dependência dá-se o nome de *Dependência Transitiva.* Cada dependência transitiva dá origem a uma nova tabela na 3FN. Quando nos confrontamos com dependência de dependências

aconselha-se começar a dividir a tabela pela coluna que possui menor número de dependências.

No exemplo corrente, as colunas, *nome_paciente e residência_paciente* não dependem do *cod_consulta* mas sim directamente do *cod_paciente* que, no entanto, não é chave da tabela. Estas colunas estão transitivamente na dependência do cod_paciente. Por ai então, dá-se o desdobramento da tabela A' em duas outras ficando a tabela original A, dividida da seguinte forma:

A_1'(*cod_consulta, cod_paciente, data_consulta)*

A_2'(*cod_paciente, nome_paciente, residência_paciente)*

A_1'' (cod_consulta, *cod_exame)*

A_2'' (cod_exame, *descriacao_exame)*

Deste modo, a tabela A ficou Normalizada na 3FN.

Convém observar que, se uma tabela esta na 3FN ela, também, está na 1FN e 2FN.

2.4. Linguagem formal de consulta para modelos relacionais.

Uma linguagem de consulta é aquela através da qual o usuário solicita informação à base de dados. Estas linguagens podem ser classificadas em procedimentais e não procedimentais. As procedimentais são aquelas nas quais o usuário instrui o sistema no sentido de levar a cabo uma série de operações na BD com a finalidade de calcular resultados desejados. Nas linguagens não procedimentais, pelo contrário, o usuário descreve as informações desejadas sem especificar um procedimento concreto para obter estas informações.

Normalmente destacam-se três linguagens puramente de consulta, a saber: A álgebra relacional que é procedimental, cálculo relacional de tuplas (registos) e cálculo relacional de domínios, sendo as duas últimas *não procedimentais*. Estas linguagens são rígidas e formais, pelo que a maior parte dos sistemas comerciais de base de dados relacionais oferecem linguagens de consulta mistas, que para além de serem ricos em sintaxe, oferecem, adicionalmente, sub-linguagens de definição de dados, administração de segurança e outras.

Seguidamente, debruçaremos sobre uma das linguagens puras de consultas à base de dados – a Álgebra Relacional, com o objectivo de apresentar uma base teórica para introduzir a Linguagem Estruturada de Consulta ou SQL (Structured Query Language). Deixamos pois, de fora, o cálculo relacional.

2.4.1. Álgebra Relacional

Podemos definir a álgebra relacional como um conjunto de operações que são necessárias efectuar a fim de manipular relações. Estas operações são provenientes, em parte, da teoria de conjuntos – ramo da Matemática. Qualquer operação executada sobre uma relação (tabela) dá origem a uma nova relação que pode, também, ser manipulada. São definidos operadores à semelhança de "+" e "-" da álgebra usual para actuar nas tabelas e obter os resultados desejados.

A Álgebra Relacional é pois, uma linguagem de interrogação procedimental, visto que o utilizador dá instrução para o sistema executar uma sequência de operações na base de dados, e calcular o resultado esperado.

2.4.1.1. Funções da Álgebra Relacional

Estão definidas, na álgebra relacional, nove operações que passamos a listar em três grupos:

- Union – União ($\cup$).
  Intersection– Intersecção ($\cap$).
  Difference – Diferença, Subtracção (-).
  Product – Produto ($\times$).

  } *Provenientes da Teoria dos Conjuntos*

- Project– Projecção($\boldsymbol{\pi}$);
  Select– Selecção($\boldsymbol{\sigma}$);
  Join– Junção ($\triangleright\triangleleft$);
  Divide – Divisão ($\div$).

  } *Aplicam-se especificamente ao modelo de dados relacionais*

- Assignment – Designação, Atribuição – Operação padrão das linguagens computacionais.

Para explicar a sintax de cada uma dessas funções, vamos nos suportar em pequenos exemplos que, em alguns casos, sairá de um estudo de caso a ser desenvolvido ao longo deste trabalho (parte prática).

a) Operação União

Sejam duas relações *R* (A_1, A_2, ..., A_n) e *S* (B_1, B_2, ..., B_n), onde A_i e B_i são atributos de R e S respectivamente, definidas de tal forma que:

- *R* tem mesma aridade que *S*
- *Dom*(A_i) = *Dom*(B_i)

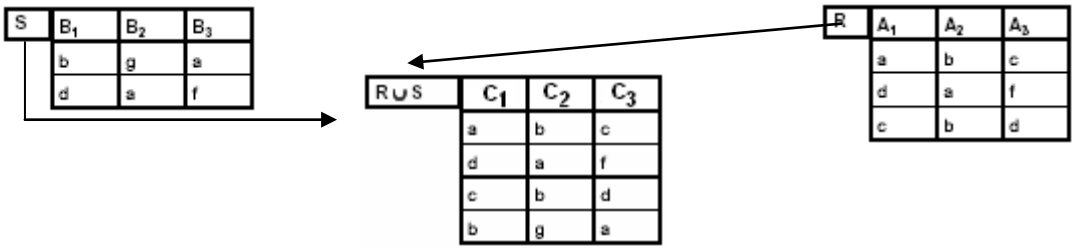
Define-se *R* união *S* como

$R \cup S = U(C_1, C_2, ..., C_n)$, Com mesma aridade *que* ***R*** *e* ***S***;

Dom (Ai) =Dom (Bi) = Dom (Ci).

A relação *U,* resultante, contem todas as tuplas de R ou S sem repetição.

Esta operação é útil na reconstrução de relações fragmentadas (ver secção 4.2 – fragmentação de dados) por conveniência de distribuição de dados.



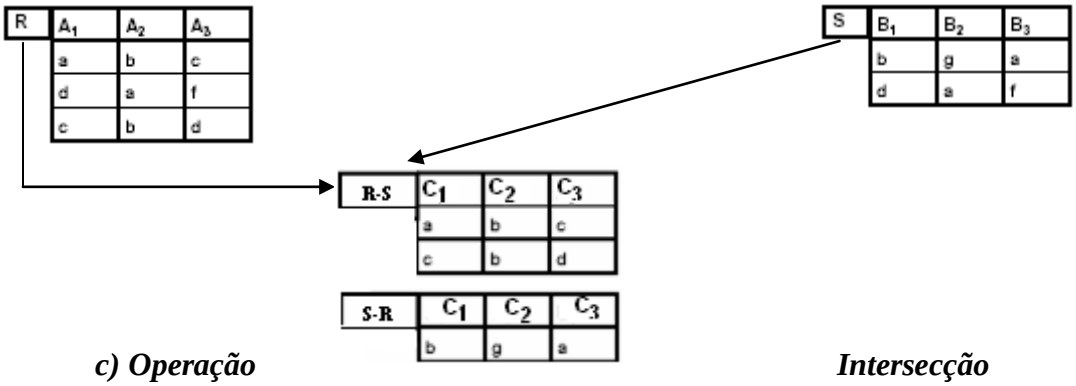
b) Operação Diferença (R-S)

$R - S = D(C_1, C_2, ..., C_n)$, Onde *D* tem mesma aridade e R e S;

Dom (Ai) =Dom (Bi) = Dom (Ci).

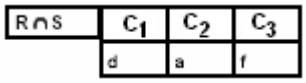
D contem todas as tuplas de R que não são de *S.*

Observemos que a operação diferença é não comutativo, ou seja R-S ≠ S-R



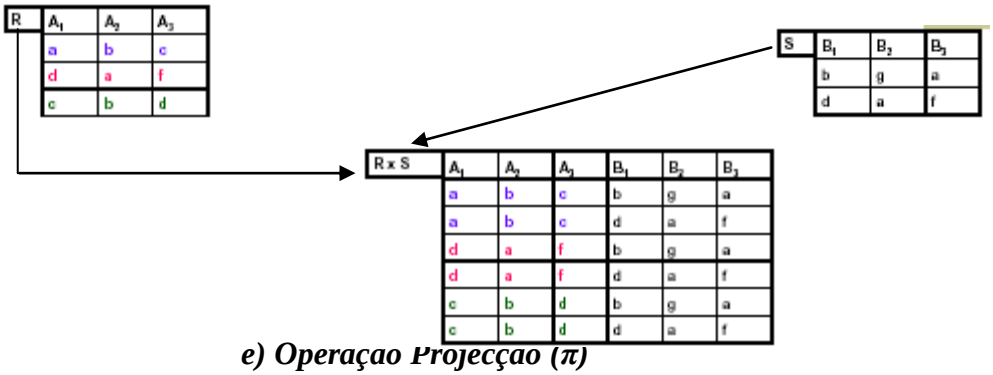
c) Operação Intersecção

$R \cap S = R - (R - S) = I(C_1, C_2, ..., C_n)$. I contem tuplas contidas, simultaneamente nas relações operandas R e S. No exemplo em decurso, a relação intersecção de R e S tem apenas uma tupla,



d) O Produto Cartesiano (sobre relações)

Sejam as relações *R* (A_1, A_2, ..., A_n) e *S* (B_1, B_2, ..., B_m), o produto cartesiano *A*×*S* define-se como uma outra relação *P=A*×*S*, de *n+m* atributos contendo todas as tuplas resultantes da combinação de cada tupla de R com todas de S.



e) Operaçao Projecçao (π)

Seja *R* uma relação e *X* um subconjunto de atributos de *R* ($X \subset Att(R)$). Define-se Projecção de *R* no conjunto de atributo *X*, simbolizado por $\pi_X(R)$, como uma relação que se obtêm reduzindo o domínio de cada tupla de R a um número de atributo escolhido para X. A projecção é uma operação unária.

Por exemplo: pretende-se exibir apenas o código do paciente, o nome e o grupo sanguíneo da relação Paciente. Simbolicamente ficava: $\pi_{codP,nomeP,GsangP}(Paciente)$

codP	***nomeP***	***GsangP***
AF1701756373	Gertrude Melo	RH+
SM1506782356	Cristofen Jangada	RH-

f) Operação Selecção (σ)

A operação Selecção é também unária. Transforma uma relação em outra e é indexada por um predicado envolvendo valores das tuplas.

Notado por $\sigma_P(R)$, onde:

σ – Símbolo de selecção;

P – predicado, como sendo um termo lógico ou uma função boleana aplicada a argumentos que podem ser também predicados;

R – a relação sobre a qual recai a operação.

Deste modo, escrever $\sigma_P(R)$ significa seleccionar todas as tuplas da relação R que satisfazem a condição definida em P.

Por exemplo, se estivéssemos interessados em exibir todos os pacientes internados cujas idades são superior a 10 e não superior a 20, escrevíamos:

$$\sigma_{10<idadeP<=20}(P\_\text{int}ernados)$$

g) Junção (> <)

Em álgebra relacional, uma junção de um atributo x duma relação R com y de S, produz o conjunto de todas as tuplas t, tal que t é a concatenação de uma tupla r pertencente a R e uma tupla s pertencente a S que satisfaça o predicado "Rx θ Sy é verdadeiro" (sendo θ o operador relacional e os atributos Rx e Sy pertencendo mesmo domínio). Simbolicamente: $R><_{(Rx\theta Sy)}S$.

O resultado desta operação é utilizado para combinar tuplas relacionadas de duas relações em tuplas simples:

R	A_1	A_2	A_3
	a	b	c
	d	b	c
	b	b	f
	c	a	d

S	B_1	B_2
	a	d
	b	e

$R \rhd\lhd_{A2>B1} S$	A_1	A_2	A_3	B_1	B_2
	a	b	c	a	d
	d	b	c	a	d
	b	b	f	a	d

Notemos que a junção considera apenas as combinações de tuplas que satisfazem a condição de junção. Esta é a principal diferença entre o produto cartesiano e a Junção. Naquela todas as combinações de tuplas são incluídas no resultado.

É muito comum o caso de junção cuja condição envolve apenas a comparação de igualdade. Neste caso, a junção é denominada *Equijoin*.(ver exemplo a seguir)

$R \rhd\lhd_{A2=B1} S$	A_1	A_2	A_3	B_1	B_2
	a	b	c	b	e
	d	b	c	b	d
	b	b	f	b	e
	c	a	d	a	d

Podemos ver que, há sempre um ou mais pares de atributos com valores idênticos em todas as tuplas (A_2=B_1). Para por cobro a esta duplicação desnecessária, é criada uma outra operação, a ***Junção Natural***. Esta, nada mais é senão uma Equijoin seguida da remoção de todas duplicações(os atributos desnecessarios). Denota-se por *. O exemplo antrior escrito com a operação Junção Natural seria:

$R *_{(A_2=B_1)} S$, com resultado no quadro ao lado: (foi mantido, no resultado, o atributo da primeira relação, R).

$R *_{A2=B1} S$	A_1	A_2	A_3	B_2
	a	b	c	e
	d	b	c	d
	b	b	f	e
	c	a	d	d

Nota: O conjunto $\{\sigma, \pi, \cup, -, \times\}$ é um conjunto completo, isto é, quaisquer outras operações da álgebra relacional pode ser expressas como uma sequência de operações deste conjunto. Com efeito, a Intersecção, por exemplo, pode ser expressa usando União e Diferença:

$$R \cap S \equiv (R \cup S) - [(R - S) \cup (S - R)]$$

Demonstração:

Preliminarmente, vamos demonstrar três igualdades da teoria dos conjuntos.

Sejam os conjuntos A, B e C. A' representa o complementar do conjunto A.

$$i)\ (A - B) - C = A - (B \cup C)$$
$$ii)\ A - (B - C) = A \cap (B' - C)$$
$$iii)\ (A \cup B) \cap (A \cup B') = A$$

Com efeito:

$$i)\ (A - B) - C = (A \cap B') \cap C' = A \cap (B' \cap C')$$
$$= A \cap (B \cap C)' = A - (B \cup C); \qquad (A - B = A \cap B')$$

$$ii)\ A - (B - C) = A \cap (B \cap C')' = A \cap (B' - C)$$

$$iii)\ (A \cup B) \cap (A \cup B') = [(A \cap A) \cup (A \cap B')] \cup [(B \cap A) \cup (B \cap B')]$$
$$= [A \cup (A \cap B')] \cup [(B \cap A) \cup \varnothing]$$
$$= [A \cup (A - B)] \cup [(B \cap A)]$$
$$= A \cup (B \cap A)$$
$$= A$$

Voltemos agora a $R \cap S \equiv (R \cup S) - [(R - S) \cup (S - R)]$.

$$\begin{aligned}(R\cup S)-\left[(R-S)\cup(S-R)\right] &= \left[(R\cup S)-(R-S)\right]-(S-R) \quad \textit{por i)}\\ &= \left[(R\cup S)\cap\left(R'\cup S\right)\right]-(S-R); \quad \textit{por ii) ou } \left(R-S=R\cap S'\right)\\ &= S\cap\left(S'\cup R\right)=\left(S\cap S'\right)\cup\left(S\cap R\right) \quad \textit{por iii)}\\ &= \varnothing\cup\left(S\cap R\right)=\left(S\cap R\right), \textit{ como queriamos demonstrar}\end{aligned}$$

A operação *Junção* pode ser especificada através de um *Produto Cartesiano* seguido de *Selecção*: $R><_{(Rx\theta Sy)} S\equiv\sigma_{(Rx\theta Sy)}(R\times S)$. Da mesma forma, a Junção Natural é expressa com ajuda do produto cartesiano seguido pelas operações *Selecção e Projecção.*

Como se pode aperceber, as diferentes formas de junção como a intersecção "não são estritamente necessárias para expressar o poder da álgebra relacional...". "No entanto, elas são muito convenientes porque são frequentemente utilizadas pelas aplicações de base de dados relacionais"[5]. A operação Divisão é também uma de outras operações introduzidas por conveniência.

A operação Divisão é aplicada para duas relações R(Z) ÷ S(X), onde X $\subseteq$ Z. Seja Y=Z - X; isto é, Y é o conjunto de atributos de R que não são atributos de S. O resultado da divisão é uma relaçãoT(Y) que incluirá uma tupla t se tR cujo tR[Y] = t aparecer em R com tR[X] = Ts para toda tupla tS em S. Isto significa que para uma tupla aparecer no resultado T da divisão, os valores em t devem aparecer em R em combinação com todas as tuplas de S.

A operação de divisão pode ser expressa como uma sequência de operações π, χ e -:

$$T_1=\pi_y R$$
$$T_2=\pi_y((S\times T_1)-R)$$
$$T=T_1-T_2$$

[5] - INTRODUÇÃO A BANCO DE DADOS, TAKAI, Osvaldo Kotaro et all, 2005, DCC IME USP

2.4.1.2. Aninhamento de operações

Grande parte das consultas à base de dados requer combinação de varias operações. É pois, possível compor operações mais complexas da álgebra relacional, através de aninhamento de operações mais simples. Essa possibilidade existe porque as operações, em álgebra relacional, actuam sobre relações devolvendo, como resultado, também relações sobre as quais podem actuar novamente.

Exemplo 1: *alista a idade de todos pacientes internados com hepatite*. O resultado pode ser obtido pela composição das funções selecção e projecção como mostramos a seguir: $\pi_{P\_Idade}(\sigma_{Diagnostic=Hepatite}(P\_\text{int}ernados))$. Observa que a operação de selecção gera uma relação como resultado que foi tomada como relação de entrada para a operação de projecção.

Exemplo 2:*Listar nomes e morada de todos os pacientes que estiveram presentes na consulta do dia D*. Esta tarefa pode ser executada por partes:

1) $\text{Re}l_1 = Paciente \times Consulta\,(produto\,cartesiano\,entre\,as\,duas\,relaçoes)$
2) $\text{Re}l_2 = \sigma_{P\_codP=C\_codP}(\text{Re}l_1)$
3) $\text{Re}l_3 = \sigma_{C\_data=D}(\text{Re}l_2)$
4) $\text{Re}l_4 = \pi_{P\_nome,P\_morada}(\text{Re}l_3)$

Numa só expressão seria: $\pi_{P\_nome,P\_morada}(\sigma_{C\_data=D}(\sigma_{P\_codP=C\_codP}(Paciente \times Consulta)))$. Equivalentemente podia ser escrito usado a Junção no lugar do Produto cartesiano seguido de selecção. Escrever-se-ia: $\pi_{P\_nome,P\_morada}(\sigma_{C\_data=D}(Paciente >< _{P\_codP=C\_codP} Consulta))$

Nem todas as consultas podem ser expressas em álgebra relacional. Algumas destas consultas são conhecidas como Funções Agregadas sobre uma colecção de valores da base de dados. São elas: SOMA, MÉDIA, MÁXIMO e MÍNIMO. Para contagem de tuplas usa-se a função

CONTAR. Um outro tipo de consulta muito utilizado é a que envolve agrupamento de tuplas de uma determinada relação pelo valor de alguns dos seus atributos.

Uma operação FUNCTION fica bem definida da seguinte forma:

$\langle atributo\ de\ agrupamento \rangle \Im_{\langle lista\ de\ funçoes \rangle}(nome\ da\ relaçao)$,onde <atributos de agrupamento> é uma lista de atributos da relação especificada em <nome da relação> e <lista de funções> é uma lista de pares (<função><atributo>). Em cada par, <função> é uma das funções seguintes: SOMA, MÉDIA, MÁXIMO e MÍNIMO, CONTAR, e <atributo> é um atributo da relação. Relação resultante tem tantos atributos quanto aqueles que existirem em atributos de agrupamento além da lista de funções combinada. Por exemplo, para recuperar o número de internados por cada enfermaria e obter a média de suas idades escreve-se:

$\langle seccao\_entern \rangle \Im_{\langle COUNT\ cod\_entern, AVERAGE\ idade \rangle}(Enternamento)$

2.5. SQL (structured query language) – Linguagem Estruturada de Consulta

O SQL foi desenhado e implementado pela IBM Research e rapidamente se tornou "um padrão de linguagem de consultas comerciais"[6], em particular para as bases de dados relacionais. Usa uma combinação de construtores da Álgebra e Cálculo Relacional.

O facto de ser chamado linguagem e consulta não significa que só serve para isso. Na verdade, o SQL possibilita os programadores a escrita de códigos nas aplicações de base de

[6] - INTRODUÇÃO A BANCO DE DADOS, TAKAI, Osvaldo Kotaro et all, 2005, DCC IME USP

dados, dando lugar ao que se chama *SQL embebido*. Pode ser usado como o Data Definition Language – DDL (Linguagem de Definição de Dados) ou como Data Manipulation Language – DML (Linguagem de Manipulação da Dados) já que possui recursos que permitem: Definir dados; Consultar dados; Actualizar dados. Além do mais, conta com mecanismos de definição de vistas e criação /eliminação de índices.

O interesse subjacente ao desenvolvimento deste capítulo é tão-somente apresentar o SQL como uma linguagem de consulta, mostrando sua estrutura e compara-lo à Álgebra relacional com o fito de justificar o conhecimento desta como um dos suportes teóricos na fundamentação do SQL.

2.5.1. Estrutura básica duma consulta em SQL

A estrutura básica de uma consulta em SQL é suporta-se em três cláusulas, a saber: *SELECT*, *FROM* e *WHERE*.

A cláusula SELECT corresponde à operação Projecção da Álgebra relacional, e é utilizado para indicar quais os atributos se quer exibir no resultado da consulta.

A cláusula FROM desempenha o papel que a operação Produto Cartesiano tem na Álgebra Relacional. Permite indicar quais as relações intervêm na consulta.

A cláusula WHERE corresponde ao predicado da operação selecção da álgebra relacional. Consiste em um predicado envolvendo atributos das relações que afiguram na cláusula FROM. Nesta cláusula, ainda, podem aparecer expressões aritméticas envolvendo os operadores de comparação <, >, =, >=, <=, e <> e operandos constantes ou atributos das tuplas. Há lugar, também, para os conectores lógicos AND, OR e NOT.

Uma consulta típica em SQL esta na forma:

$SELECT\ A_1, A_2, ..., A_n$
$FROM\ R_1, R_2, ... R_m$, onde A_i são os atributos das R_i relações envolvidas na consulta e P é
$WHERE\ P$
o predicado ou condição da consulta.

Esta consulta equivale à seguinte expressão da Álgebra Relacional

$$\pi_{A_1, A_2, ..., A_n}(\sigma_P(R_1 \times R_2 \times ... \times R_m)).$$

A lista de atributos pode ser substituído por *, quando se quer exibir todos os atributos de todas as relações envolvidas. A cláusula WHERE pode ser omitida e, neste caso, a consulta assume o predicado como sendo verdadeiro alistando, no resultado, todas as tuplas de todas as relações envolvidas.

Como é de se esperar, o resultado de uma consulta SQL é assumida por ele como uma relação com a qual ele pode voltar a operar. No entanto, nestas relações podem afigurar tuplas repetidas o que não acontece em Álgebra Relacional, pois, aqui uma relação é equivalente a um conjunto, e, o conceito de conjunto descarta a existência de elementos repetidos. Todavia, quando se quer um resultado SQL isento de repetições, faz-se seguir a clausula SELECT da palavra DISTINCT.

O SQL é, relacionalmente, tão potente quanto a Álgebra Relacional. Demonstraremos este facto mostrando as equivalentes, em SQL, às cinco operações básicas da Álgebra Relacional:

i) para concretizar uma reunião de relações, o SQL utiliza o operador UNION. Com efeito, sejam as relações R_1 e R_2, $R_1 \cup R_2$ escreve-se da seguinte forma:

```
SELECT *
  FROM  R1
UNION
SELECT *
  FROM  R2
```

ii) a Diferença R_1-R_2 se concretiza utilizando o predicado EXISTS que devolve um valor verdadeiro se existe relação que satisfaz uma determinada subconsulta:

```
SELECT *
 FROM R1
WHERE NOT Exists
(SELECT *
 FROM R2
WHERE R1.A1 = R2.A1,...., R1.An = R2.An)
```

onde A_i são atributos de R_1 ou R_2.

iii) O produto Cartesiano $R_1 \times R_2$ se realiza com a própria clausula *from* como já foi dito anteriormente:

```
SELECT *
 FROM R1, R2
```

iv) A Selecção $\sigma_P(R_1)$ é concretizado na clausula WHERE:

```
SELECT *
 FROM R1 , onde P é predicado indicando a condição que as tuplas devem satisfazer.
WHERE P
```

v) A projecção, $\pi_{A_i}(R_1)$, é feito logo na clausula SELECT:

```
SELECT Ai
 FROM R1
```
, onde A_i é a lista dos atributos de R_1, a projectar.

Claro está, que as outras operações da Álgebra Relacional, também, possuem equivalências em SQL. Alias, foi mostrado, na secção anterior, que quaisquer outras operações da AR pode ser expressa pela combinação das operações básicas, logo expressáveis em SQL. Contudo, algumas palavras reservadas do SQL conseguem expressar de forma directa tais operações. É o caso de INTERSECT para a intersecção, INER JOIN para Junção etc..

Em SQL, também, é possível o cálculo de funções em grupo de tuplas. Isto é conseguido utilizando a clausula *GROUP BY* da seguinte forma: o grupo é formado de acordo com todos os atributos que aparecem na cláusula *GROUP BY*, e cada tupla que devolve o mesmo valor em cada um deste atributos são colocados no mesmo grupo.

São ofertadas, pelo SQL, funções para calcular:

- a média: AVG
- A soma / Total: SUM
- o máximo: MAX
- O mínimo: MIN
- A contagem: COUNT.

Estas operações são chamadas *Funções de Agregação*. Operam sobre conjunto de tuplas e devolvem um único valor. Vejamos um exemplo da estrutura:

Observe que, a entrada para as operações AVG e SUM devem, obrigatoriamente, ser conjunto de números.

$$
\begin{aligned}
&SELECT\ \ A_2, MAX(A_3)\\
&\ FROM\ \ R_1 R_2 ... R_m\\
&WHERE\ \ P\\
&GROUP\ BY\ A_2;
\end{aligned}
$$

Sobre o resultado de um agrupamento é possível estabelecer condições que actuará a nível de grupo, e não a nível de tuplas como acontece na cláusula *WHERE*. Estas condições são estabelecidas na cláusula *HAVING* e podem ser usadas funções de agregação.

$$
\begin{aligned}
&SELECT\ \ A_2, AVG(A_3)\\
&\ FROM\ \ R_1 R_2 ... R_m\\
&WHERE\ \ P \qquad\qquad , c \in ¡\\
&GROUP\ BY\ A_2\\
&HAVING\ \ AVG(A_3) > c;
\end{aligned}
$$

É importante salientar que, quando se utiliza agrupamento, na clausula SELECT só podem aparecer atributos pelos quais se processam tal agrupamento e/ou as funções de agregação.

Muito utilizados são as ***Consultas Aninhadas*** – as chamadas *Sub-consultas.* Usualmente aplicadas a: (1) membros de conjuntos, (2) comparação de conjuntos e a (3) verificação de relações vazias. Vejamos a estrutura de cada uma das aplicações:

(1) Membros de conjuntos

A consulta aninhada é utilizada para verificar se uma tupla é membro ou não de uma relação. O conectivo *in* testa os membros de um conjunto, no qual este conjunto é a colecção de valores produzidos na clásula select.

$$
\begin{aligned}
&SELECT\ \ A_1\ A_2, ...A_n) \\
&\ FROM\ \ R_1\ R_2\ ...R_m \\
&WHERE\ \ P\ \mathbf{in}\ (Select\ A_1\ A_2\ ...A_n \\
&\qquad\qquad FROM\ R_1\ R_2\ ...R_m \\
&\qquad\qquad WHERE\ P);
\end{aligned}
$$

(2) Comparação de conjuntos

A consulta aninhada permite usar comparações do tipo > ***some*** (maior que pelo menos uma), **<= *some***, **= *some***, etc., ou > ***all*** (maior que todas), **>= *all***, etc.

$$
\begin{aligned}
&SELECT\ \ A_1\ A_2, ...A_n) \\
&\ FROM\ \ R_1\ R_2\ ...R_m \\
&WHERE\ \ P > \mathbf{all}\ (Select\ A_1\ A_2\ ...A_n \\
&\qquad\qquad FROM\ R_1\ R_2\ ...R_m \\
&\qquad\qquad WHERE\ P);
\end{aligned}
$$

(3) Verificação de relações vazias

A consulta aninhada permite testar se o resultado de uma subconsulta é vazio. O construtor ***exists*** devolve o valor verdadeiro se o argumento da subconsulta for não-vazio.

$$
\begin{aligned}
&SELECT\ \ A_1\ A_2, ...A_n) \\
&\ FROM\ \ R_1\ R_2\ ...R_m \\
&WHERE\ \ P\ \mathbf{Exists}\ (Select\ A_1\ A_2\ ...A_n \\
&\qquad\qquad FROM\ R_1\ R_2\ ...R_m \\
&\qquad\qquad WHERE\ P);
\end{aligned}
$$

2.5.2. Exemplos de consultas em SQL

1. Número de pacientes atendidos, por dia, a partir de uma determinada data (*uma consulta com parâmetro*):

SELECT *Count(**C**.cod_consulta) AS [Nr de Atendimento], **C**.data_consulta*
FROM *Consulta as **C***
WHERE ***C**.data_consulta>[Insira a data a partir da qual se quer contar]*
GROUP BY ***C**.data_consulta;*

2. Nº de casos de doenças diagnosticadas:

SELECT ***D**.cid, **N**.causa, Count(C.cod_consulta) AS [Nº de Caso]*

FROM *Nosologia AS **N**, Consulta AS **C**, Diagnostico AS **D***

WHERE *(((C.cod_consulta)=[D].[cod_consulta]) and (N.cid)=[D].[cid])*

GROUP BY *D.cid, N.causa;*

3. Nº de internados por cada secção de internamentos acompanhada da média das idades:

SELECT *seccao_entern AS Secção, COUNT(cod_enternamento) AS [Nº de enternados], Avg(idade) AS [Media das idades]*

FROM *Paciente AS **P**, Consulta AS **C**, Enternamento AS **E***

WHERE *P.cod_paciente=C.cod_paciente And E.cod_consulta=C.cod_consulta*

GROUP BY seccao_entern;

4. Nº de casos de doenças distribuído por sexo e por idade. – *Consulta de referencia cruzada:*

TRANSFORM *Count(D.idade) AS ContarDeidade*

SELECT *D.cid, D.causa, D.sexo, Count(D.sexo) AS [Total sexo]*

FROM *C_casos_doença AS D*

GROUP BY *D.cid, D.causa, D.sexo*

PIVOT *D.idade;*

3. Base de dados centralizados (única) versus base de dados distribuídos, sincronizados por replicação

Uma das grandes preocupações de qualquer sistema de gestão de base de dados, prende-se com a disponibilidade dos dados.

Diversas soluções tecnológicas têm sido desenvolvidas e implementadas, cada uma adaptada à realidade em estudo e, por isso, exigindo recursos diferentes para sua implementação.

Tentaremos uma abordagem, não exaustiva, de bases de dados centralizadas e distribuídas com o fito de tornar evidente os aspectos que concorrem para o maior ou menor encarecimento de suas implementações.

Um sistema centralizado caracteriza-se, essencialmente, pela existência de um único servidor que contem a base de dados e estações clientes que acessam, remotamente, às aplicações do servidor. Sistemas com estas características acarretam custos de comunicação consideráveis, pois, o sistema requer ligação permanente entre as estações clientes e o servidor central. Essa ligação poderá se concebida de duas formas (ambas custosas), quais sejam:

1. Por linhas ou circuitos, alugados de um provedor de serviço de dados, entre cada estação cliente e o servidor central (fig.3-1), ou

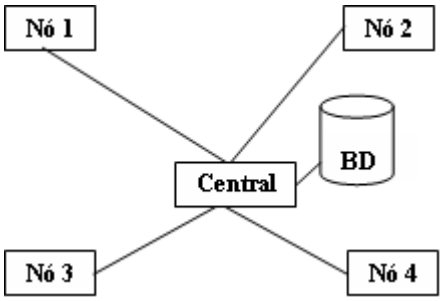


Fig. 3-1 - Topologia em estrela de uma BD centralizada

2. Via Internet, com uma conexão dedicada do sistema central à internet, requerendo todo um aparato de segurança que exige algum esforço financeiro. Igualmente, as estações clientes devem estar permanentemente conectadas ao sistema central, através da internet, enquanto trabalham (fig.3-2).

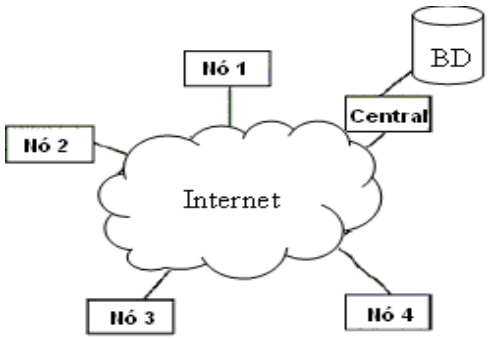


Fig. 3-2- Base de dados Centralizada via Internet

Como podemos deduzir, se, a nível das estações clientes, falhar a conexão, estas ficam impossibilitadas que trabalhar, pois não dispõem de dados localmente para o efeito.

Dado ao ambiente de negocio cada vês mais competitivo em que as empresas têm que actuar, elas vêem-se obrigadas a distribuírem geograficamente ajustando estratégias locais mais favoráveis à produção e/ou exploração do seu negocio. A necessidade dessa distribuição das instalações, aliada à pretensão de garantir alguma autonomia local em concomitância com a diminuição de custo de comunicação e sem pôr de lado a necessidade de manter uma perspectiva integrada dos dados da empresa/organização, justifica a migração de um sistema centralizado para um sistema de base de dados distribuídos.

Podemos definir um sistema de base e dados distribuídos como um conjunto de nós, cada um mantendo, localmente, uma base de dados, que estão logicamente integrados a através de redes de computadores, possibilitando o acesso aos dados de maneira transparente (fig.3-3).

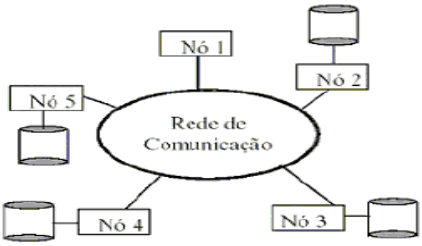


Fig. 3-3 - Base de dados Distribuídos

3.1.Vantagens de bases de dados distribuídos

A descentralização organizacional, o processamento económico bem como maior autonomia estão de entre os motivos para se optar por uma base de dados distribuídos. Estas motivações encontram fundamentos em alguns princípios que passamos a descrever e que, de certa forma, constituem em promessas da tecnologia SBDDs (Sistema de Base de Dados Distribuídos):

a) Aumento do desempenho

De entre os elementos que concorrem para o aumento do desempenho das BDD, destacamos: a *disponibilidade* – a falha de um nó contendo a relação r não resulta na indisponibilidade de r se existirem réplicas, o que reduz a demora pelo acesso remoto aos dados; *o paralelismo no processamento de consultas* – uma consulta sobre r pode ser processada por vários nós em paralelo; a *redução de dados a transferir* – uma relação r está disponível localmente em cada nó que contenha uma réplica de r.

b) Confiabilidade através de transacções distribuídas

As transacções distribuídas são executadas em vários nós acedendo ao banco de dados local. Assim, com suporte total para transacções distribuídas, os aplicativos podem acessar uma única imagem lógica do banco de dados e confiar no SGBD distribuído para assegurar que suas solicitações sejam executadas correctamente, independentemente do que acontecer no sistema [ÖZSU, 1999].

c) Facilidade de expansão

Um sistema distribuído pode crescer mais facilmente que um sistema centralizado. Se for necessário expandir o sistema porque o volume de dados cresceu ou o volume de processamento aumentou, é mais fácil acrescentar um novo nó à rede de computadores, desde que os nós sejam autónomos, do que substituir um sistema centralizado já existente por outro maior.

3.2.Formas de distribuição de dados

Um dos objectivos deste trabalho é propor a replicação como uma das estratégias de distribuir dados, a ser desenvolvido nos capítulos 4 e 5. No entanto, queríamos fazer referência à fragmentação de dados. De facto, tem-se, muitas vezes, a necessidade de dividir uma relação *R* em unidades lógicas menores para fins de distribuição, o que chamamos fragmentação de dados.

Se uma relação *R* está fragmentada, então está dividida em um número *n* de fragmentos (f_1, f_2,...., f_n) de tal forma, que estes contêm informações suficientes para a reconstrução da relação original. Esta reconstrução pode ser feita aplicando a operação de união ou um tipo especial de junção dos vários fragmentos (*ver secção 2.4.1.1. - funções de álgebra relacional*). Uma relação pode ser fragmentada horizontalmente ou verticalmente.

Fragmentação Horizontal: cada registo/tupla é atribuído a um ou mais fragmentos, prevalecendo a lógica de que tais fragmentos devem estar no nó onde o registo é mais frequentemente acedido. Por exemplo a relação paciente do HRSN pode ser assumida no Tarrafal ou S. Miguel como fragmentos, contendo, por isso, apenas os registos dos pacientes dos respectivos concelhos. Um fragmento horizontal pode ser definido como uma selecção de tuplas sobre uma relação global R:

$$R_i = \sigma_P(R)$$

A reconstrução da relação R pode ser obtida tomando a união de todos os fragmentos, isto é:

$$R = R_1 \cup R_2 \cup .... \cup R_n$$

Fragmentação Vertical: neste caso o próprio registo é dividido em partes que são guardadas em nós diferentes, onde cada parte é mais frequentemente acedida. Nesta fragmentação, o atributo chave deve estar presente em todos os fragmentos ou então é atribuída a cada registo uma identificação (atributo) que permite a junção eficiente dos fragmentos. Esta fragmentação é exactamente a *Operação Projecção* vista em *Funções da Álgebra Relacional:*

$$r_j = \pi_{R_1,R_2,...R_m}(R),\ m < n$$

A reconstrução de R pode ser obtida tomando a Junção Natural de todos os fragmentos, isto é:

$$R = r_1 \otimes r_2 \otimes ... \otimes r_n$$

3.3.Custo de centralização

Ficou já implícito que os sistemas centralizados, geralmente, acarretam mais custos de que os sistemas distribuídos. Todavia, neste capítulo, procuraremos apoiar em dados concretos a fim de nos convencermos deste facto.

Antes de prosseguirmos, vamos descrever a realidade que tomamos como caso para estudo:

A Região Sanitária Santiago Norte (RSSN), que é uma experiência piloto em matéria de regionalização dos serviços de saúde em Cabo Verde, cobrindo, territorialmente, a área correspondente aos municípios de Santa Catarina, Santa Cruz, São Lourenço dos Órgãos, São Miguel, São Salvador do Mundo e Tarrafal. A sede da região situa-se em Achada Falcão no concelho de Santa Catarina, a uma distância média de 13,2 km (medidas feitas em linha recta) em relação às estruturas de saúde correspondentes nos outros concelhos.

O nosso objectivo é propor a utilização de um modelo de comunicação de dados entre os diferentes municípios e a sede da região.

Claro está que, a realidade já é geograficamente distribuída, pelo que, qualquer modelo teria que atender a esta especificidade. Seguidamente vamos apresentar quatro vias para a implementação de comunicação de dados, todas elas compatíveis com a distribuição

geográfica da realidade em estudo e, ao mesmo tempo, elaborar um plano de custo para cada uma:

Obs: consideramos suficiente a velocidade de 128 Kbps para todas as linhas de comunicação do sistema ilustrados a partir deste momento.

1) Linhas Dedicadas (Leased Lines)

Linhas alugadas de um provedor de serviços de comunicação possibilitando uma ligação directa entre dois pontos. Particularmente, no caso em estudo, entre a estrutura de cada concelho e a sede da RSSN.

É um recurso fortemente recomendado a sistemas que requerem operações e resultados em tempo real.

Se optarmos por esta via de comunicação de dados, teremos a seguinte representação gráfica (Fig. 3-4):

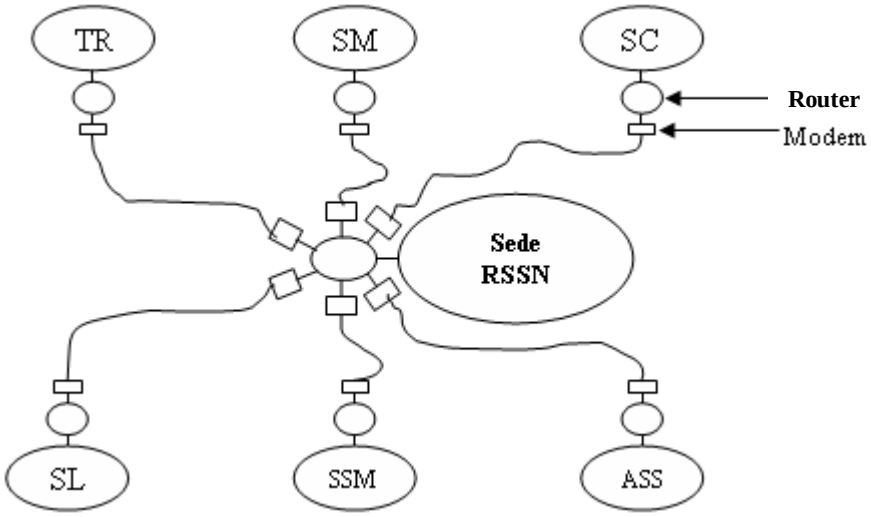


Fig. 3-4 - Modelo de comunicação de dados por Linhas Dedicadas

Custos:

a) Custos de instalação

a1: aquisição de equipamentos

Equipamentos	Quantidade	Preço Unit.	Total
Modem 128 Kbps	12	60.000,00	720.000,00
Router 1 porta serie	6	120.000,00	720.000,00
Router 6 porta serie	1	300.000,00	300.000,00
		Total	**1.740.000,00**

Tabela 3-1 – Custo de equipamento para Linha Dedicada

a2: instalação de equipamentos

Instalação e Programação dos equipamentos	**250.000,00**

Tabela 3-2 – Custo de programação de equipamento Linha Dedicada

a3: custos de instalação pelo provedor[7]

Descrição	Quantidade	Preço Unit.	Total
Circuitos Locais (DSSC) [8]	1	5.000,00	5.000,00
Outros circuitos de 128K	5	52.000,00	260.000,00
		Total	**265.000,00**

Tabela 3-3 – Custo de instalação pelo provedor de Linha Dedicada

b)Custo de funcionamento

b2: custo de assinatura mensal

Descrição	Quantidade	Preço Unit.	Total
Cicuitos Locais (DSSC)	1	9.400,00	9.400,00
Circuito Intra-Ilha de 128K	5	160.000,00	800.000,00
		Total	**809.400,00**

Tabela 3-4 – Custo mensal Linha Dedicada

[7] - Todos os valores foram calculados, em ESC.CV, mediante informações recolhidas perto do provedor. Vide quadros constantes do anexo.
[8] - Delegacia de Saúde de Santa Catarina

2) Circuitos Dedicados (Frame Relay)

A diferença desta com a primeira (Leased Lines) é que nesta a conexão de cada par não é feita directamente. Os postos ligam-se à uma rede Frame Relay do provedor e cabe, então, a este criar as conexões (circuitos) virtuais dos pares (fig. 4-6).

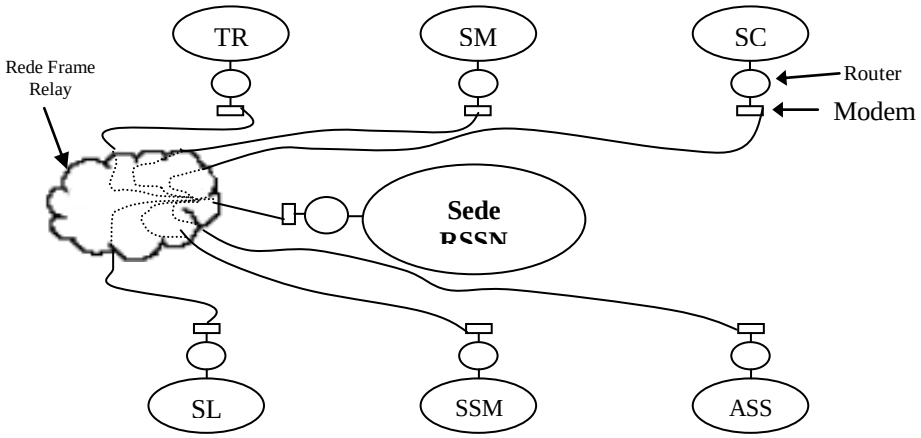


Fig. 3-5- Modelo de comunicação de dados por circuitos dedicados

Custos:

a) Custos de instalação

a1: aquisição de equipamentos

Equipamentos	Quantidade	Preço Unit.	Total
Modem 128 Kbps	6	60.000,00	360.000,00
Modem 1Mbps	1	100.000,00	100.000,00
Router 1 porta serie	7	120.000,00	840.000,00
		Total	**1.300.000,00**

Tabela 3-5 – Custo de equipamento para circuito dedicado Frame Relay

a2: instalação de equipamentos

Instalação e Programação dos equipamentos	**200.000,00**

Tabela 3-6 – Custo de programação de Equipamentos suporte do Frame Relay

a3: custos de instalação pelo provedor

Descrição	Quantidade	Preço Unit.	Total
Acessos de 128K	6	60.000,00	360.000,00
Acessos de 768K	1	80.000,00	80.000,00
		Total	**440.000,00**

Tabela 3-7 – Custo de Instalação Frame Relay pelo provedor

b)Custo de funcionamento

b2: custo de assinatura mensal

Descrição	Quantidade	Preço Unit.	Total
Acessos de 128K	6	110.700,00	664.200,00
Acessos de 768K	1	290.320,00	290.320,00
Circuitos lógicos de 128K	6	128.450,00	770.700,00
		Total	**954.520,00**

Tabela 3-8 – Custo mensal de circuito dedicado Frame Relay

3) Via Internet com a tecnologia VPN – Virtual Private Network

"A rede privada virtual (VPN, virtual private network) consiste numa extensão de uma rede privada que abrange ligações encapsuladas, encriptadas e autenticadas entre redes públicas ou partilhadas. As ligações VPN podem facultar ligações de acesso remoto e encaminhadas a redes privadas na Internet"[9].

Cada posto terá um acesso a Internet através do ADSL - Asymmetric Digital Subscriber Line que é o tipo de ligação à Internet mais comum e acessível ao grande público. O sistema central (sede RSSN) estará ligado a Internet através duma linha dedicada. O VPN, constituído pelo sistema central e pelos diferentes postos, permitira o estabelecimento de circuitos lógicos permanente e seguros entre eles.

A comunicação de dados por esta via far-se-ia de forma permanente (on line) entre os postos e a sede da RSSN.

[9] In centro de ajuda e suporte Windows XP Professional

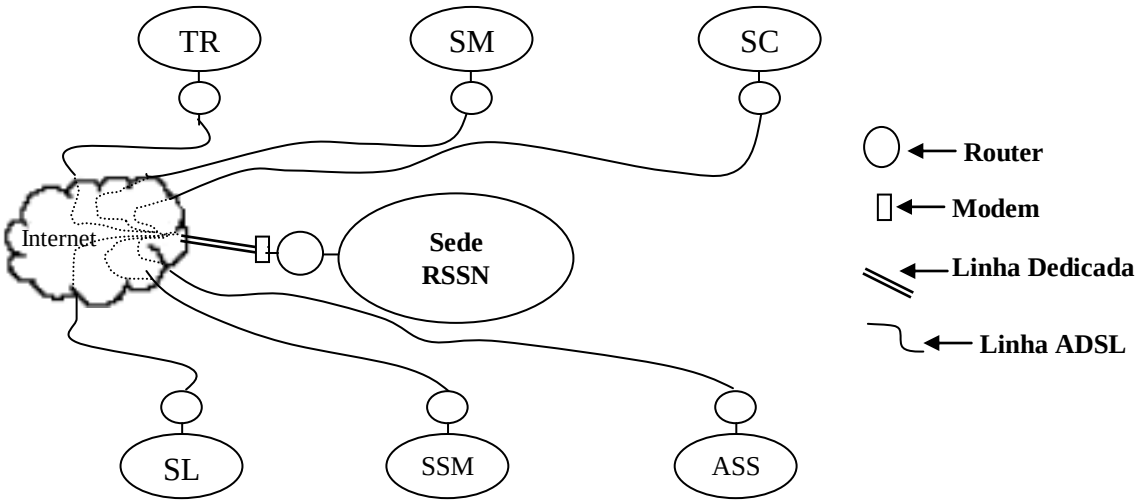


Fig. 3-6 – Modelo de comunicação de dados via Internet (ADSL)

Custos:

a) Custos de instalação

a1: aquisição de equipamentos

Descrição	Quantidade	Preço Unit.	Total
Router / Modem ADSL	6	5.900,00	35.400,00
Modem 128 K	1	60.000,00	60.000,00
Router com 1 porta Serie	1	120000	120.000,00
		Total	**215.400,00**

Tabela 3-9 – Custo de equipamento VPN

a3: custos de instalação pelo provedor

Descrição	Quantidade	Preço Unit.	Total
Instalação / programação dos equipamentos (ADSL)	6	3.000,00	18.000,00
Instalação / programação dos equipamentos (L. dedicada	1	60.000,00	60.000,00
		Total	**78.000,00**

Tabela 3-10 – Custo de instalação VPN

b)Custo de funcionamento

b1: custo de assinatura mensal

Descrição	Quantidade	Preço Unit.	Total
Assinatura mensal ADSL	6	3.000,00	18.000,00
Assinatura mensal Linha dedicada	1	180.000,00	180.000,00
		Total	**198.000,00**

Tabela 3-11 – Custo de assinatura mensal VPN

4) Via Internet com replicação

A comunicação de dados por esta via far-se-ia numa lógica de sincronização periódica de réplicas da base de dados situadas nos diferentes pontos (no caso em estudo, seria municípios ou estruturas de saúde).

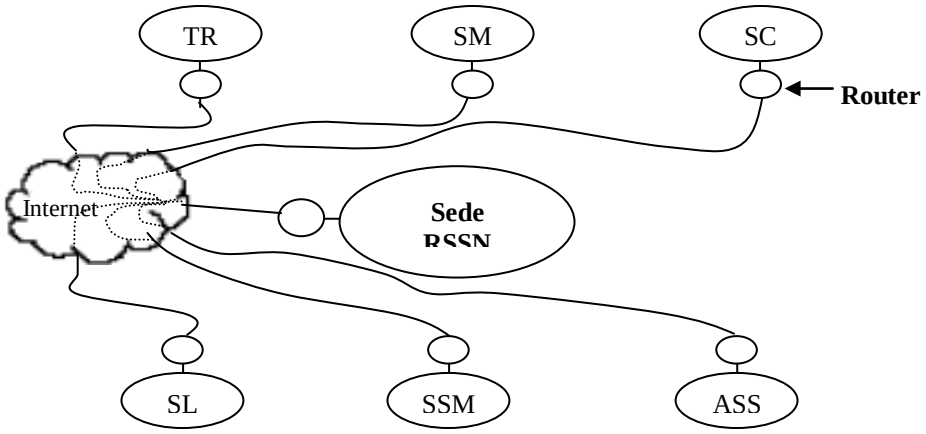


Fig. 3-7 – Modelo de comunicação de dados via Internet (ADSL)

Custos:

a) Custos de instalação

a1: aquisição de equipamentos

Descrição	Quantidade	Preço Unit.	Total
Router / Modem ADSL	7	5.900,00	41.300,00

Tabela 3-12 – Custo aquisição de equipamentos para comunicação via ADSL com replicação

a2: Instalação e programação de equipamentos

Descrição	Quantidade	Preço Unit.	Total
Instalação / programação dos equipamentos	7	3.000,00	21.000,00

Tabela 3-13 – Custo Instalação de equipamentos *para* comunicação via ADSL com replicação

b)Custo de funcionamento

b1: custo de assinatura mensal

Descrição	Quantidade	Preço Unit.	Total
Assinatura mensal	7	3.000,00	21.000,00

Tabela 3-14 – Custo de assinatura mensal *para* comunicação via ADSL com replicação

Face a estes dados e, de acordo com as quatro opções de ligação, se quisermos uma solução centralizada, com servidor na sede da RSSN e com a necessidade de actualização em tempo real, teremos que fazer a 1ª, 2ª ou 3ª opção suportando um custo muito elevado. Reparemos que, a terceira opção, a menos custosa de entre as três primeiras, requer um investimento na ordem dos 461.400,00 para a instalação incluindo uma mensalidade de 198.000,00. Contudo, a realidade da RSSN e o propósito para o qual se criamos a base de dados – Sistema de Gestão de Dados Estatísticos do HRSN, não carece de actualidade em tempo real. Desta forma, a opção mais ajustada, porquanto responde as nossas necessidades e incrivelmente menos cara que as duas primeiras opções, seria a ultima opção.

4. Replicação de base de dados

A definição mais simples e básica que se pode atribuir à replicação seria a de uma cópia. Todavia as regras e os resultados que definem esta cópia determina a sua diferença de uma simples cópia de segurança. Copiar consiste em ler de uma origem e escrever em outro lugar - destino, exactamente o que foi lido. Replicar, por outro lado, pode ser entendido como copiar uma selecção de dados e disponibilizá-los para um ou vários destinos, de forma progressiva e contínua, ou seja, a medida que novas informações são adicionadas, retiradas ou modificadas na origem, assim também será feito nos destinos, ao mesmo tempo ou com a periodicidade definida pelo ABD (Administrador da Base de Dados).

4.1.Tipo de replicação

Atendendo ao *tempo de actualização* das réplicas, podemos tipificar uma replicação de *Síncrona* e *Assíncrona*. No primeiro caso, quando alguma cópia do banco é alterada, essa alteração será imediatamente aplicada a todos os outros bancos dentro da transacção. A Replicação Síncrona é apropriada em aplicações comerciais onde a consistência exacta das informações é de extrema importância. Contrariamente, na *replicação assíncrona*, se um BD é alterado, a alteração será propagada e aplicada para outro BD num segundo momento, dentro de uma transacção separada sendo que esta poderá ocorrer segundos, minutos, horas ou até dias depois. A cópia poderá ficar temporariamente fora de sincronia, mas quando a sincronização ocorrer os dados convergirão para todos os locais especificados.

Tendo em conta a direcção do fluxo de dados, define – se replicação mestre-escravo e multi-mestre. A *Replicação mestre-escravo* permite atualizações em somente uma das réplicas

chamada *primária.* As demais réplicas (*Escravas*) recebem as modificações através da cópia primária (*mestre*) após sincronização. Na replicação multímetro a actulizaçao é feita a partir de qualquer das réplicas.

Ainda, pode-se falar de *replicação Homogénea* ou *Heterogénea*. Nas replicações homogéneas, todas as bases de dados envolvidas são geridas pelo mesmo produto (SGBD). Nas replicações heterogénea, múltiplos SGBD são envolvidos.

4.2. Quando e Porquê Replicar?

A opção por uma base de dados replicados passa, essencialmente, pela natureza da própria realidade a modelar. Na verdade, esta decisão é tomada após se ter respostas a perguntas como: quantos postos vão utilizar tal BD? Como estão, geograficamente, distribuídos? Precisarão, todos eles, de dados actualizados em tempo real?

Se diferentes postos geograficamente dispersos vão partilhar a mesma BD e sem a necessidade de actualizações de dados em tempo real, temos ai então, um cenário adequado para uso de replicação. Tal uso concorre para a economia de recursos financeiros que suportariam, doutra forma, a comunicação de dados entre os diferentes postos.

4.3. Mecanismos de gestão da replicação

Trabalhar com BDs replicados pressupõe um percurso por três etapas essenciais: a criação de Réplicas, a Sincronização de dados e a Resolução de Conflitos decorrentes da sincronização. Cada SGBD tem seus mecanismos de replicação. Para além disso, existem softwares desenvolvidos especificamente para conduzir o processo de replicação (Replicadores), com particularidade de, alguns deles, replicar para SGBDs diferentes. Como exemplos, temos:

- *SQLServer Replication*: possibilita replicação mestre-escravo e multi-mestre com actualização Síncrona ou Assíncrona;

- *O Replicação de Dados 2.0*: é um replicador com código fonte em Delphi e permite criar tantas conexões quanto se quer com as base de dados: DB2, Interbase/Firebird, MS Access, MSSQL (Microsoft SQL Server), MySQL, Oracle, Dbase, Paradox.

- *Daffodil Replicator*: é um replicador para Postgresql, Oracle, SQLServer, Daffodil DB etc.. é um open Source (grátis) para aplicações GPL mas, pago para aplicações comerciais. Permite replicação assíncrona Multi-Mestre e entre diferentes SGBDs;

- *Object Multi-Master Replication Server*: é um replicador para Postgresql, Oracle, SQLServer, Sybase, DB2, etc. Replica entre diferentes SGBDs.

Muitos outros softwares poderíamos listar aqui, contudo, o que nos interessa mesmo são os mecanismos de replicação que existe à disposição de quem trabalha com MS Access quais sejam:

- *Menus internos do Access*: oferece a maioria dos recursos suportados pelo Jet[10] 4, e também uma ferramenta visual para solução de conflitos. É simples de se usar, mas requer o Access instalado;

- *O Porta Arquivos do Windows* (Windows Briefcase), muito simples mas bastante limitado. (o utilizador só precisa arrastar e largar o arquivo);

- *Replication Manager:* é uma ferramenta visual disponível a quem tem a versão Developer do Microsoft Office. Oferece diversos recursos de replicação e sincronização, sem precisar ter o Access instalado na máquina.

- *Código JRO (Jet and Replication Object):* é o principal e mais completo recurso de programação disponível ao desenvolvedor para se trabalhar com replicação. É usado para automatizar soluções de forma a não precisar da intervenção do usuário, e nem do Access instalado na máquina. Não oferece mecanismo já pronto para resolução de conflitos, mas é possível codificar um.

[10] Jet Engine é o motor de Gestão de Base de dados que o MS Access utiliza.

- *Código DAO:* é outro recurso de programação disponível ao desenvolvedor, mas não é tão completo quanto o JRO.

4.4. Problemática da replicação / Sincronização (gestão de conflitos de sincronização)

Uma etapa fulcral de todo o processo de replicação é, pois, a resolução de conflitos decorrentes do processo de sincronização - processo que permite actualizar dois membros de um conjunto de réplicas, trocando todos os registos actualizados em cada membro. Diz-se que dois membros do conjunto de réplicas estão sincronizados quando as alterações efectuadas em cada membro tiverem sido aplicadas ao outro membro. Durante este processo ocorrer diversas situações que podem ocasionar conflitos. Seguidamente apontaremos algumas desta situações seguido de proposta de resolução:

a) Conflito de chave única

Um conflito de chave única pode ocorrer em uma das formas a seguir:

- Duas réplicas inserem um novo registro com o mesmo valor em um campo que é uma chave primária ou tem uma chave única.

- A Estrutura Mestre cria um índice único e uma réplica simultaneamente adiciona dois ou mais registros com o mesmo valor da chave. Quando a alteração do esquema chega à primeira réplica com registros chave duplicados, então, o primeiro registro chave duplicado permanece na tabela base.

Proposta de resolução: se as chaves forem numeração automática, uma solução possível seria fixar intervalos de valores para cada réplica. No entanto, no nosso trabalho prático, para as chaves mais sensíveis como é o caso de código do paciente e

código consulta, implementaremos funções (código VB) que, de forma automática, combinam as 1as strings da localização da réplica e uma sequência de número. Desta forma, teremos códigos únicos em cada réplica e no conjunto.

b) Conflito de validação em nível da tabela

Um conflito de validação em nível de tabela ocorre quando são inseridos dados que violam uma regra de validação em nível de tabela que restringe os valores ou tipos de dados que podem ser inseridos em uma tabela.

Proposta de resolução: Impedir, a nível dos controlos nos formulários, a introdução de dados que violem a regra de validação.

c) Conflito de Actualização de integridade referencial

Um conflito de actualização de integridade referencial ocorre quando a chave primária é actualizada em uma réplica e novos registros filho que fazem referência ao valor original da chave primária são adicionados a uma réplica diferente.

Proposta de resolução: no nosso caso este conflito não ocorre porquanto, os códigos, chaves primárias, são gerados automaticamente por funções de modo que os utilizadores não poderão actualiza-los para valores diferentes.

d) Conflito de exclusão de integridade referencial

Um conflito de exclusão de integridade referencial ocorre quando um registro de chave primária é excluído em uma réplica enquanto novos registros filho que fazem referência à chave primária excluída são adicionados em uma segunda réplica.

Proposta de resolução: podemos evitar este conflito impedindo exclusão a nível das réplicas (ver secção 5.1). Todavia, se esta medida for excessivamente restritiva, então

podemos optar em minimizar o risco de surgimento deste conflito impondo restrições a nível de grupo de utilizadores.

e) Conflito de bloqueio

Um bloqueio de conflito ocorre quando o registro não pode ser aplicado durante a sincronização porque outro usuário bloqueou a tabela.

Proposta de resolução: este conflito não terá lugar no nosso modelo, na medida em que a sincronização acontece na sede da RSSN com cópia das réplicas, enviadas pelo correio electrónico dos diferentes concelhos. Garante-se, ali, que estas cópias não estarão abertas durante a sincronização. Logo não existirá conflitos de bloqueio.

f) Conflito de violação da chave externa

Um conflito de violação de chave externa ocorre quando há um registro de chave primária inválido. Isso pode ser causado por qualquer um dos outros tipos de conflitos.

Proposta de resolução: evitando os conflitos anteriores.

5. Replicação no Microsoft Access – estudo de caso **(projecto de gestão de informações estatística para a Região Sanitária Santiago Norte)**

Como já vimos (secção 4.3), existem vários SGBDs com ferramentas para implementação da replicação. É o caso do Oracle, o Postgre SQL, o SQL Server etc. A razão de termos escolhido o Microsoft Access preende-se com o facto deste ser o mais acessível e de fácil implementação em comparação com as outras soluções.

O Access pode ser compartilhado numa estrutura: Servidor de Ficheiro (File Server) que consiste em dividir a aplicação em front-end (interfaces com o utilizador) e back-end (dados), de forma que cada estação rode o motor Jet e acesse o arquivo de dados centralizado no servidor; ou Cliente – Servidor (Client Server), em que o Access é usado como front-end de um banco de dados SQL Server, sendo que todos os dados são geridos pelo servidor.

A Replicação é a terceira forma de compartilhar uma base de dados Jet. Esta estrutura permite que cada utilizador trabalhe de forma isolada na sua própria réplica de BD e, com alguma periodicidade, os dados são sincronizados actualizando as cópias.

O Access dispõe de recursos próprios que permitem a criação, a sincronização de réplicas e a resolução de conflitos.

5.1. A criação das Réplicas

A situação que propomos modelar (RSSN), tem, já instituído, um modelo de fluxo de dados (estatísticos). Cada concelho, parte da região, deve enviar dados estatísticos para o gabinete técnico da região situada em Santa Catarina e este, fará tratamento e posterior envio para os serviços centrais com uma periodicidade mensal.

A sede da RSSN deve, em determinado momento, ter dados actualizados dos diferentes concelhos. Por isso, propomos uma topologia estrela com replicação Multi-Direcional na qual a *Estrutura Mestre ou Réplica de estrutura Global* (cópia onde é possível a actualização da estrutura) fica instalada na sede da RSSN e as demais réplicas (as cópia sem acesso ao modo estrutura) nos outro concelhos (fig.5-1).

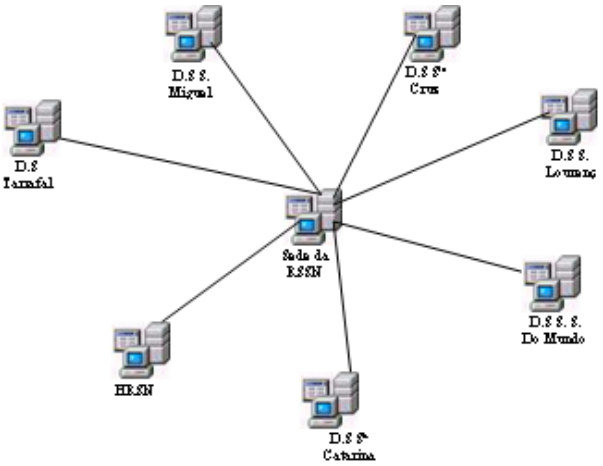


Fig. 5-1 – Topologia em estrela de replicação Multi-direcional

Para converter a base de dados bd.mdb em estrutura mestre, abrimo-la em seguida clicamos no menu Ferramentas → Replicação → Criar Réplicas. O assistente de replicação do Access pede a confirmação e informa que, em caso de confirmação, a base de dados será fechada a fim de ser criada a réplica e ao mesmo tempo converter-se-á em réplica de estrutura global (fig.5-2).

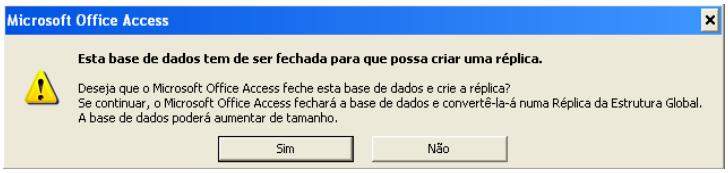


Fig. 5-2 – Janela do assistente de replicação

Seguidamente, nos é recomendado a efectuar uma cópia de segurança. O assistente faz isso automaticamente ao respondermos afirmativamente o apelo (fig.5-3):

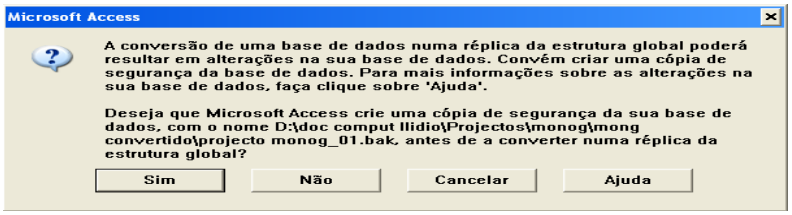


Fig. 5-3 – Janela do assistente de replicação

E, é criado imediatamente a réplica, ficando para nós a tarefa de guarda-la com o nome e em lugar desejado. Também, o assistente nos dá a hipótese de, antes de guardar, definirmos propriedades como a Prioridade e a opção de impedir eliminações (Fig. 5-4):

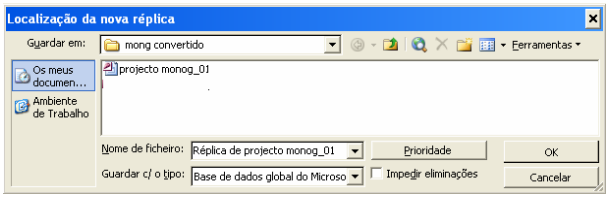


Fig. 5-4 – Janela do assistente de replicação

Não ocorrendo erros, receberemos uma mensagem de êxito e a Réplica de Estrutura Global (Estrutura – Mestre) será aberta. E, observemos que as tabelas replicadas, apresentam ícones de replicação (fig. 5.5).

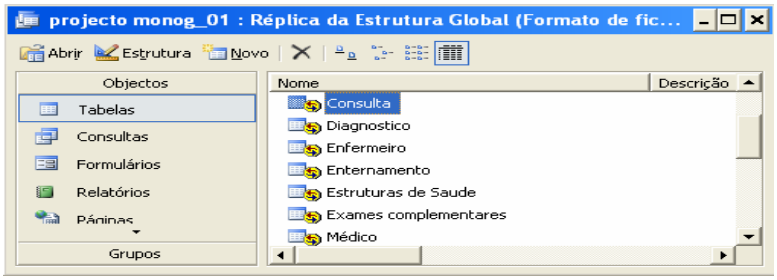


Fig. 5-5– Janela do assistente de replicação

Cada tabela e consulta tem a propriedade Replicável, podendo haver no MDB objectos replicáveis e objectos locais. Para alterar esta propriedade, basta clicar com o botão direito do rato sobre o objecto (na tela principal do MDB) e abrir a janela de propriedades. Convém, contudo, observar que tal alteração só e possível nas Réplicas de Estrutura Global que são passíveis de replicação.

Antes de criar outras réplicas é importante estabelecer as propriedades que estas devem ter. Nomeadamente devemos certificar:

- Se será R*éplica Total* ou P*arcial*. Será total se nenhum objecto da Estrutura Global for desmarcado a propriedade Replicável, portanto, todos os objectos são replicados. Do contrário, obtêm-se Réplicas Parciais;

- Se, aos utilizadores da réplica a criar serão permitidos a exclusão de registos ou não. Para isso, deixamos desmarcada ou marcamos a caixa "Impedir Eliminações" (Fig.5-5).

- Que *visibilidade* pretendemos para a réplica. Cada réplica pode ter três níveis de visibilidade: global, local ou anónimo. A **Réplica Global** pode ser sincronizada com outras réplicas globais e com as réplicas geradas a partir dela própria. A **Réplica Local**, por sua vez, pode ser sincronizada apenas com a réplica global que a gerou. Por fim as **Réplicas Anónimas**, as quais funcionam de maneira semelhante às locais, porém suas informações não ficam permanentemente armazenadas na tabela Replicada. É útil para réplicas que raramente são sincronizadas, e também para cenários envolvendo Internet.

- Que *prioridade* terá a réplica a criar, no conjunto de replicação. No motor de base de dados **Jet 4**, Cada réplica recebe um nível de prioridade (entre 0 e 100), que será usado para resolver os *conflitos de sincronização.* Por predefinição, a Estrutura Global Réplica de Estrutura Global recebe o nível 90, e cada réplica recebe 90% da réplica que a gerou. Em caso de empate, a réplica com o menor ID (ou seja, a mais antiga) vence. O jet 4 implementa, ainda, a *Prioridade histórica.* Por exemplo: consideremos três réplicas, R_1, R_2 e R_3 com prioridades 90,80 e 70 respectivamente e com conflito de actualização num determinado registo. Quando R_1 é sincronizada com R_3, a vencedora será R_1, pois, tem maior prioridade. No entanto, se logo de seguida R_3 for sincronizada com R_2, pelo conceito de prioridade histórica, sairá vencedora a Réplica R_3 mesmo tendo menor prioridade que R_2, pois o registo veio de R_1 que tem maior prioridade. Desta forma, fica claro que, independentemente da ordem de sincronização, prevalecerá sempre a réplica de maior prioridade.

5.2. Sincronização em Access

O Microsoft Access, como foi dito anteriormente, possui ferramentas próprias para a sincronização de réplicas bem como para a resolução de conflitos que podem surgir ao longo do processo. Todavia, a existência destas ferramentas não inviabiliza a opção de criarmos, via código, procedimentos de sincronização e resolução de conflitos.

O JRO, Jet and replication object, nos permite, usando VBA, escrever códigos para controlar a sincronização de um membro com o resto do conjunto de réplica. Só é utilizado para ficheiros.**mdb** nas versões do Access 2000 ou posteriores. Para Microsoft Access 97 ou anterior, tem de utilizar *Data Access Objects* (DAO).

Para sincronizar um ficheiro.mdb usando as ferramentas do Microsoft Access, o assistente nos orientam conforme os passos seguintes:

- Abrir a réplica de estrutura global (outra réplica do conjunto de replicas)[11], clicar em ferramentas, de seguida apontar para replicação e clicar em sincronizar agora. Abrir-se-á a janela que mostramos na fig.5-6 na qual devemos indicar o caminho da réplica com a qual pretendemos sincronizar.

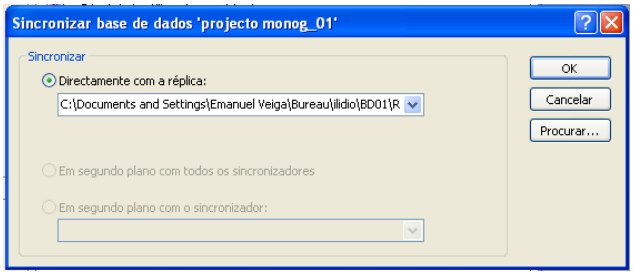


Fig. 5-6 – Janela do assistente de replicação

- Ao clicar OK é exibida uma janela que nos informa da necessidade de fecharmos a base de dados para o efeito de sincronização e, pede confirmação (Fig.5-7).

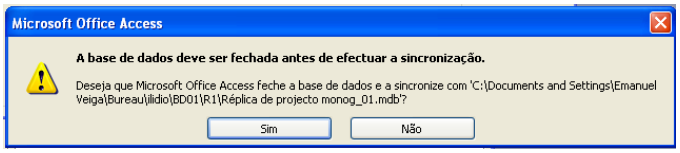


Fig. 5-7 – Janela do assistente de replicação

Confirmando, o motor executa a tarefa. Não havendo nenhum conflito, o assistente exibe a seguinte mensagem Fig.5-8:

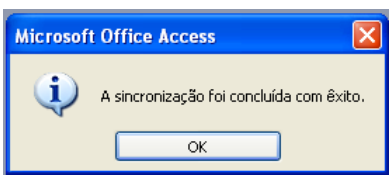


Fig. 5-8 – Janela do assistente de replicação

[11] - As réplicas locais e anónimas só sincronizam com as respectivas réplicas distribuidoras

Clicando OK, é aberto a base de dados e, podemos certificar que as actualizações foram feitas.

- Caso ocorra algum conflito, seja o de actualização, então uma mensagem alusiva é mostrada (Fig.5-9):

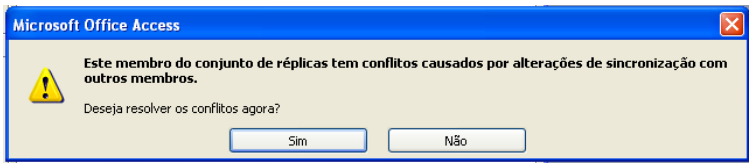


Fig. 5-9 – Janela do assistente de replicação

E pergunta se queremos resolver o conflito. Se clicarmos sim, será mostrada a janela Visualizador (fig.5-10) de conflitos do Microsoft Access, na qual podemos ver as tabelas que contêm conflitos. Seleccionando uma tabela e clicando no botão visualizar, são exibidos os campos desta tabela, podendo, opcionalmente, ter visível todos eles ou, apenas os campos em conflito (fig.5-11). Para além disso, esta janela contempla uma breve descrição da razão do conflito. Os campos em conflitos nas duas réplicas, são mostrados lado a lado, sob o título "conflito preferido" e "conflito preterido".

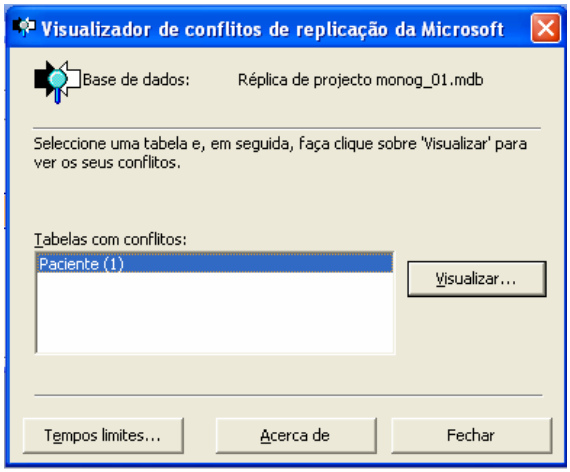


Fig. 5-11– Visualizador de conflitos de replicação

Na coluna de conflito preferido, o botão "Manter alteração dominante", quando clicado, o conflito é resolvido prevalecendo as alterações constantes nestes campos. De outro modo, restam as opções, "adiar resolução" ou "Resolver com estes dados". Esta ultima opção, faz com os dados a permanecer na base de dados sejam os da coluna "conflitos preterido" ao invés dos da coluna "conflito

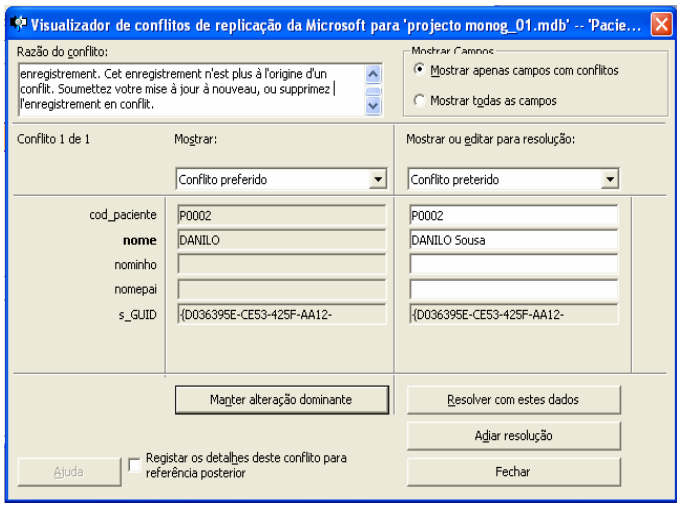


Fig. 5-10– Visualizador de conflitos replicação (detalhes)

preferido" que, por defeito, são dominantes.

O campo "S_GUID" (Globally Unique Identifier), guarda valor que identifica a linha de forma única entre as réplicas, mesmo que o valor da chave primária mude.

6. Conclusão

Como um trabalho monográfico que é, não podia deixar de constituir em mais um momento de aprendizagem e consolidação de conhecimentos adquiridos ao longo da formação.

Ao longo deste trabalho ficamos convencidos de que a implementação de uma solução centralizada ou distribuída depende, em primeiro lugar, de a realidade ser ou não distribuída e da necessidade de dados actualizados em tempo real. E, face a uma realidade distribuída e sem necessidade crítica de actualização, as soluções distribuídas por replicação tendem a ser incrivelmente menos custosos do que as soluções centralizadas.

De facto, a *replicação* não é apenas uma alternativa para instituições com fracos recursos financeiros, conforme diz o título deste trabalho, mas sim uma opção inteligente, uma oportunidade de reduzir custos de comunicação quando a realidade da organização permite sua implementação.

A componente prática deste trabalho permitiu-nos, de entre outras experiências, uma confrontação segura entre a álgebra relacional, fundamentada em teorias matemáticas e o SQL, linguagem de consulta a base de dados. E duvidas não ficaram, que esta encontra seus fundamentos naquela.

No decorrer deste trabalho, também, podemos perceber que é necessário uma boa base matemática no campo da álgebra de conjuntos e lógica de predicados para um conhecimento teórico mais profundo de conceitos inerentes a base de dados relacionais. Estas percepções, apesar de não terem sido contemplados como objectivos neste trabalho, constituem-se em perspectivas de investigações futuras como recomendação pessoal enquanto autor deste trabalho.

7. Bibliografias:

– GETZ Ken et al. Access 2000 Developer's Handbook. Vol.1, Desktop Edition. Sybex. 1999

– LOUREIRO Henrique. Access Macros & VBA, Lisboa. FCA – Editora Informática. Abril 2006;

– LOURENÇO Ana, António Azevedo e Vidal de Carvalho. Desenho e Implementação de Bases de Dados com Microsoft Access XP. Editora Centro Atlântico;

– GOMES Luís et al. Fundamental do Access Xp. FCA – Editora Informática, Liboa, 5ª edição, Agosto 2001

– ALENCAR Filho Edgard. Teoria elementar dos conjuntos. ed. S Paulo: Nobel 1985

– SOUSA, Sérgio. Domine a 110% ACCESS 2000. Lisboa. FCA – Editora Informática.

– DAMAS, Luís. SQL – Structured Query Language - 6ª Edição actualizada e aumentada. FCA – Editora Informática. Maio 2005

Páginas Web:

– http://r.office.microsoft.com/r/hlidOfficeHomeFromClientWithLogging/

– http://www.devmedia.com.br/articles/viewcomp.asp?comp=2663. Acedido aos 17/05/08

– http://macine.epublish.cl/tesis/. Acedido aos 17/05/08

8. Anexos